Homework #6

[ECE30021/ITP30002] Operating Systems

Mission

- Solve problem 1
  - Solve the problems on Ubuntu Linux (VirtualBox) using vim and gcc.
- Submission
  - Submit a .tgz file containing hw6_1.c and hw6_2.c on LMS.
    "tar cvfz hw6_<*student_id*>.tgz hw6_1.c hw6_2.c"
    (Do not copy&past but type the above command)
  - After compression, please check the .tgz file by decompressing in an empty directory.
    "tar xvfz hw6_<*student_id*>.tgz"
- Due date: PM 11:00:00, May 19th

Honor Code Guidelines

- “과제”
  - 과제는 교과과정의 내용을 소화하여 실질적인 활용 능력을 갖추기 위한 교육활동이다. 학생은 모든 과제를 정직하고 성실하게 수행함으로써 과제에 의도된 지식과 기술을 얻기 위해 최선을 다해야 한다.
  - 제출된 과제물은 성적 평가에 반영되므로 공식적으로 허용되지 않은 자료나 도움을 획득, 활용, 요구, 제공하는 것을 포함하여 평가의 공정성에 영향을 미치는 모든 형태의 부정행위는 단호히 거부해야 한다.
  - 수업 내용, 공지된 지식 및 정보, 또는 과제의 요구를 이해하기 위하여 동료의 도움을 받는 것은 부정행위에 포함되지 않는다. 그러나, 과제를 해결하기 위한 모든 과정은 반드시 스스로의 힘으로 수행해야 한다.
  - 담당교수가 명시적으로 허락한 경우를 제외하고 다른 사람이 작성하였거나 인터넷 등에서 획득한 과제물, 또는 프로그램 코드의 일부, 또는 전체를 이용하는 것은 부정행위에 해당한다.
  - 자신의 과제물을 타인에게 보여주거나 빌려주는 것은 공정한 평가를 방해하고, 해당 학생의 학업 성취를 저해하는 부정행위에 해당한다.
  - 팀 과제가 아닌 경우 두 명 이상이 함께 과제를 수행하여 이를 개별적으로 제출하는 것은 부정행위에 해당한다.
  - 스스로 많은 노력을 한 후에도 버그나 문제점을 파악하지 못하여 동료의 도움을 받는 경우도 단순한 문법적 오류에 그쳐야 한다. 과제가 요구하는 design, logic, algorithm의 작성에 있어서 담당교수, TA, tutor 이외에 다른 사람의 도움을 받는 것은 부정행위에 해당한다.
  - 서로 다른 학생이 제출한 제출물 간 유사도가 통상적으로 발생할 수 있는 정도를 크게 넘어서는 경우, 또는 자신이 제출한 과제물에 대하여 구체적인 설명을 하지 못하는 경우에는 부정행위로 의심받거나 판정될 수 있다.

Problem 1

POSIX

- Search the Internet for the following functions and learn how to use message passing.
  - msgget()
  - msgsnd()
  - msgrcv()
  - msgctl()

key 인자 값에 연관된 메세지 Queue 식별자를 return
int msgget (key_t key, int msgflg)
- 성공시: return 메세지 Queue의 식별자 값
- 실패시: return -1
- key: 특수한 경우 IPC_PRIVATE로 인자를 넘겨 줄 수 있는데, 이 경우 process는 개인적인 Queue를 생성하고 process 만 사용 가능
- msgflg: 권한

→ #include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>

메세지 Queue로 데이터 전송
int msgsnd (int msqid, const void *msgp, size_t msgsz, int msgflg)
- msqid: 메세지 Queue 식별자
- msgp: 전송할 데이터
- msgsz: 전송할 데이터 크기
- msgflg: 동작 옵션
- 성공시: return 0
- 실패시: return -1

→ #include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>

메세지 Queue id의 메세지를 하나 읽고 그 메세지를 Queue에서 제거
ssize_t (int msqid, void *msgp, size_t msgsz, long msgtyp, int msgflg)
- msqid: 메세지 Queue id
- msgp: 읽어온 메세지를 msgp가 가리키는 주소에 저장 (= 읽어온 메세지)
- msgsz: msgbuf의 data 크기
- msgflg: 동작옵션
- msgtyp == 0 : Queue의 첫번째 메세지를 읽어옴
- msgtyp > 0 : 그 값과 동일한 msgtype을 갖는 메세지를 읽어옴
- msgtyp < 0 : msgtype의 절대값 이하의 가장 작은 메세지를 읽어옴
- 성공시: return 0보다 큰 값
- 실패시: return -1

→ #include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>

메세지 Queue를 제어하기 위한 system call
int msgctl (int msqid, int cmd, struct msqid_ds *buf)
- msqid: 메세지 Queue id
- cmd: 제어할 command
- cmd로 IPC_STAT or IPC_SET 사용시 전달 없으면 NULL
- 성공시: return 0
- 실패시: return -1

→ #include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>

2.3. 메시지큐에 데이타 쓰기

메시지를 보내기 위해서는 msgsnd(2) 를 사용한다. 첫번째 아규먼트는 msgget 를 통해서 얻어온 메시지큐 식별자이며, 2번째 아규먼트는 메시지큐에 넘기고자 하는 구조체, 3번째 아규먼트는 2번째 아규먼트인 구조체의 크기, 마지막 아규먼트는 메시지전달 옵션으로 봉쇄할것인지 아니면 비봉쇄로 메시지를 결정하기 위해서 사용된다.

2번째 아규먼트가 메시지큐로 전달할 메시지라고 했는데, 이것은 구조체로 전달 되며, 다음과 같은 모습을 가지게 된다.

```
1  struct msgbuf
2  {
3      long mtype;
4      char mtext[255];
5  }
```

위의 모습은 메시지 구조체의 매우 전형적인 모습으로 사실멤버변수는 필요에 따라서 얼마든지 변경될수 있다. 다만 long mtype 만이 필수요소이다.

mtype 는 메시지의 타입으로 반드시 0보다 더큰 정수이여야 한다. 우리는 이 mtype 를 각각 다르게 줌으로써, 특정 프로세스에서 특정 메시지를 참조할수 있도록 만들수 있다. 예를 들어 A 라는 프로세스가 A` 라는 메시 타입을 참조해야 하고 B 는 B` 로 참조하도록 만들어야 한다면, msgbuf 를 만들때, mtype 에 A` 은 1 B` 은 2 이런식으로 메시지 타입을 정의 하고 A 는 mtype 가 1인것을 B는 mtype 이 2인것을 가지고 가도록 만들면 된다.

attachment:queue.png

위의 그림에서 처럼 mtype 을 이용해서 자신이 원하는 메시지에만 선택 적으로 접근이 가능하다. 이특성을 잘 이용하면 매우 유용하게 사용할수 있을것이다.

msgsz 은 구조체의 크기이니 그냥 넘어가고, msgflg 에 대해서 설명하겠다. msgflg 에는 IPC_NOWAIT를 설정할수 있으며 이 값을 이용해서 봉쇄형으로 할것인지 비봉쇄형으로 할것인지 결정할수 있다. IPC_NOWAIT를 설정하면, 메시지가 성공적으로 보내지게 될때까지 해당영역에서 봉쇄(block)되며, 설정하지 않을 경우 에는 바로 return 하게 된다.

2.4. 메시지큐의 데이타 가져오기

데이타는 msgrcv(2) 함수를 이용해서 가져올수 있다. 1번째 아규먼트는 메시지큐 식별자이며, 2번째가 가져올 데이타가 저장될 구조체, 3번째는 구조체의 크기, 4번째는 가져올 메시지 타입, 5번째는 세부 조종 옵션이다.

다른것들은 굳이 설명할 필요가 없는 간단한 것들이고, 다만 4번째 메시지 타입인 msgtyp에 대해서 상세히 설명하고, msgflg 를 간단히 설명하는 정도로 넘어가도록 하겠다. 우리는 메시지를 보낼 구조체를 만들때 mtype 라는것을 정의 해서, 메시지를 분류해서 접근할수 있도록 한다는것을 알고 있다. 메시지를 가져올때는 바로 msgtyp 를 통해서 자기가 원하는 msgtyp 의 메시지 구조체에 접근할수 있게 된다. **msgtyp == 0** 메시지 큐의 첫번째 데이타를 돌려준다. **msgtyp > 0** 메시지의 mtype 가 msgtyp 와 같은 첫번째 데이타를 돌려준다. **msgtyp < 0** 메시지의 mtype 이 msgtyp 의 절대값보다 작거나 같은 첫번째 데이타를 돌려준다.

msgflg 는 msgrcv 의 메시지 가져오는 형태를 봉쇄로 할것인지 비 봉쇄로 할것인지 지정하기 위해서 사용한다. IPC_NOWAIT 를 설정할경우 가져올 메시지가 없더라도 해당 영역에서 봉쇄되지 않고 바로 error 코드를 넘겨주고 리턴한다.

[System V] Shared-Memory

- Use shared memory through attached address as ordinary memory

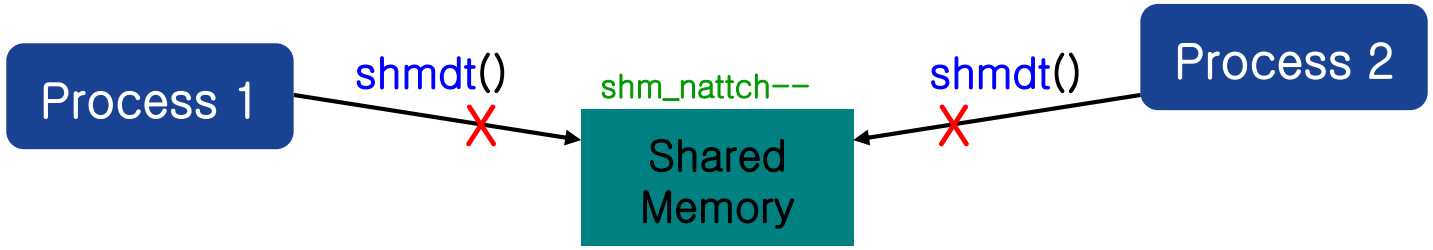
  Ex) sprintf(shared_mem, "Writing to shared memory");

- Detach shared memory from address space of process

  *int shmdt (char *shmaddr);*

  Ex) shmdt(shared_mem);

  - If all processes detaches the shared memory segment, OS discards it.



[System V] Shared-Memory

- Deallocating a shared memory block

  shmctl(shmid, IPC_RMID, NULL); →명령 실행하자마자 실행X, 예약느낌?

  - Deallocates the shared memory block when the shm_nattch becomes zero.

Reading Assignment

Search the following topics from Internet and study them

- System V shared memory
  - shmget() – allocate a share memory block
  - shmat() – attach a shared memory block to a process
  - shmdt() – detach a shared memory block from a process
  - shmctl() – controls and deallocate a shared memory block
  - www.xevious7.com/linux/lpg_6_4_4.html (Korean)
  - www.cs.cf.ac.uk/Dave/C/node27.html (English)

Problem 1

- Mission: This program reads integers from the user and accumulate them using a child process until the user inputs -1.
  - The parent and child processes communicate by messages.

- Example

```
Input integer numbers and child will accumulate (-1 to finish).
10                          // user's input
[child] input number = 10
[parent] sum = 10
20                          // user's input
[child] input number = 20
[parent] sum = 30
30                          // user's input
[child] input number = 30
[parent] sum = 60
-1                          // user's input
Parent terminating.
Consumer terminating, sum = 60
```

Problem 1

- Parent process
  - Create two message queues using msgget().
    - Parent-to-child and child-to-parent
  - Create a child process.
  - Repeat
    - Read an integer from the user.
    - Send the number to the child using use msgsnd().
    - If the input number is negative, break the loop.
    - Receive the sum of the input numbers from the child using msgrcv().
    - Display the sum. Put a prefix "[parent]" to indicate it was printed by the parent.
  - Destroy the two message queues.
  - Display a message indicating the parent is terminating.

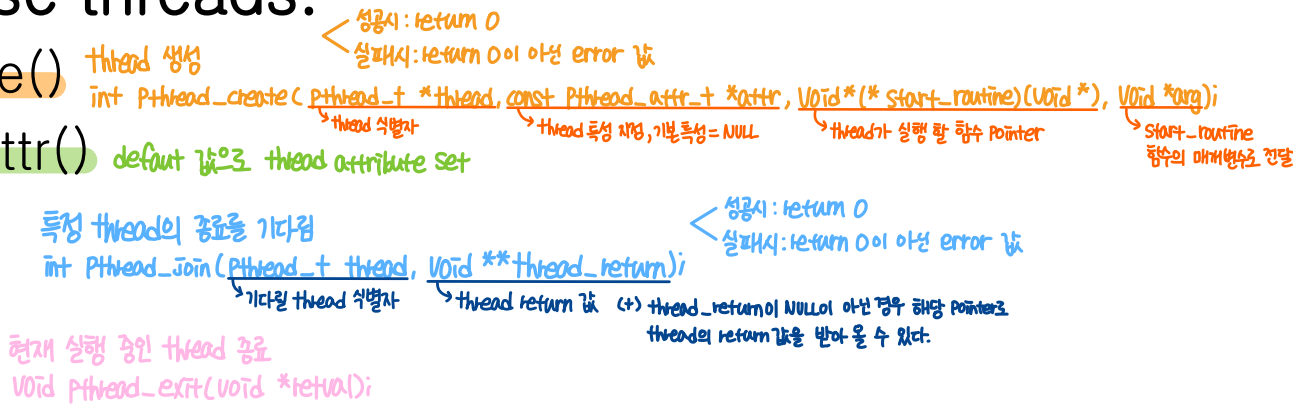
Problem 1

- Child process
  - Initialize sum by zero.
  - Repeat
    - Receive a number from the parent using msgrcv().
    - If the number is negative, break the loop.
    - Display the input number. Put a prefix "[child]" to indicate it was printed by the child.
    - Add the number to sum.
    - Send the sum to the parent using msgsnd().
  - Display a message indicating the child is terminating.

Problem 2

→ #include <pthread.h>
#include <unistd.h>

- Search the Internet for the following functions and learn how to use threads.
  - pthread_create()
  - pthread_init_attr()
  - pthread_join()
  - pthread_exit()



- You need to use '-pthread' option to use those functions.

  Ex) gcc my_code.c -pthread

Problem 2

- Mission: Compute the sum of integers from 0 to a specified number by multi-threading
  - Each thread computes the sum of i, i+m, i+2m, i+3m, …, where n is the upper bound specified by the user and m is the number of threads.
- Compilation)
  - $ gcc thread_<student_id>.c -pthread
- Examples

```
$ ./a.out 10 4          // compute the sum of integers in [0, 10] using 4 threads
info[0].sum = 12        // sum of 0, 4, 8
info[1].sum = 15        // sum of 1, 5, 9
info[2].sum = 18        // sum of 2, 6, 10
info[3].sum = 10        // sum of 3, 7
total_sum = 55          // 12 + 15 + 18 + 10

$ ./a.out 1000 10
info[0].sum = 50500
info[1].sum = 49600
info[2].sum = 49700
info[3].sum = 49800
info[4].sum = 49900
info[5].sum = 50000
info[6].sum = 50100
info[7].sum = 50200
info[8].sum = 50300
info[9].sum = 50400
total_sum = 500500
```